Rapport du projet

la réalisation d'un site d'inscriptions

Réaliser par : Fatimaezzahra Yazough
Année :2014 /2015

INTRODUCTION

Le but de ce projet et de réaliser une maquette de site web pour que les étudiant puissent s'inscrire dans différentes filières de l'université, et une commission de sélection qui va donner sont avis pour les étudiant inscrivent .

le projet est caractériser par des utilisateur qui peuvent être des étudiant ou des professeurs , ou des président de commission qui ont des rôles et des droits différents pour réaliser ce projet j'ai utiliser le framework web Django et le langage de programmation python , et finalement pour la mise en forme de pages web html et css

Table des matières

Remerciement 2
INTRODUCTION....3
CREATION DU SITE INTERNET....5
Django 5
Python 6
MODE D'EMPLOI 8
Installation de Django....8
Configuration d'une base de données....8
Commencer un projet....9
Conclusion 10

CREATION DU SITE INTERNET

Django

Django est un framework web, écrit en langage Python, et dédié à la publication sur Internet. Un framework consiste en un ensemble de bibliothèques permettant le développement rapide d'applications abouties .
Django est né en 2003 dans une agence de presse qui devait développer des sites web complets dans des laps de temps très courts (d'où l'idée du framework). Django est donc un framework Python *destiné au web*. Ce n'est pas le seul dans sa catégorie, nous pouvons compter d'autres frameworks Python du même genre comme web2py, TurboGears, Tornado ou encore Flask. Il a cependant le mérite d'être le plus exhaustif, d'automatiser un bon nombre de choses et de disposer d'une très grande communauté.

Le développement de Django a débute durant l'année 2003, dans le but de gérer efficacement plusieurs sites Internet d'actualité d'une société de presse américaine. Les développeurs de ces sites avaient besoin de pouvoir créer facilement et rapidement des sites s'interfaçant avec une base de données, ce qui donna naissance à ce framework. Développé d'abord en interne dans cette société, le framework fut finalement mis à disposition du grand public en juillet 2005, sous licence BSD, les développeurs ayant déterminé qu'il était assez mur pour une utilisation grand public. Grâce `a cette action, le framework reçoit `a l'heure actuelle beaucoup de retour d'expérience et d'améliorations de la part de la communauté. Sa version actuelle est la 0.96, il est donc toujours en développement, tout en étant totalement utilisable. Ce framework tire son nom du défunt guitariste de jazz Django Reinhardt, considéré comme le meilleur guitariste de son époque.

Django fonctionne selon un système de projets, chaque projet contenant ensuite une ou plusieurs applications (même s'il est possible d'utiliser une partie des fonctionnalités de Django sans créer d'application). Ces deux notions sont détaillées ci-après.

Un projet est une instance d'un certain nombre d'applications avec une configuration associée. La configuration, effectuée dans le fichier settings.py du projet, contient entre autre la liste des applications actives, les informations de connexion `a la base de données, ainsi que le chemin d'accès pour les templates. Le projet met aussi en place le routage des URLs par l'intermédiaire du fichier urls.py.

Une application, quant `a elle, est un ensemble de fonctionnalités Django qui sont portables (qui peuvent être utilisées dans un autre projet), comprenant généralement des modèles et des vues. Notons qu'une application peut être utilisée dans plus d'un projet, et qu'elle peut spécifier son propre routage des URLs, pour autant que le projet dans lequel elle est utilisée lui délègue ce travail.

Les champs définissent les champs requis dans la base de données. Ils sont spécifiés par des attributs de classe. Ces champs permettent de définir le type des données (entier, texte, . . .), les options du champs (null, cl´e primaire, . . .), ainsi que les relations entre les tables.

```
class Sportif(models.Model):
nom = models.CharField(maxlength=30)
prenom = models.CharField(maxlength=30)
sport = models.CharField(maxlength=30)
```

Chaque champ est représenté par l'instance d'une classe models.*Field permettant ainsi à Django de savoir quel type de donnée est contenu dans un champ. Django possède de nombreux types de données permettant d'identifier un champ, quelques uns sont détaillés ci-dessous, le lecteur se reportera `a la documentation de référence pour de plus amples détails.

Python

Afin de pouvoir appréhender au mieux Django, il convient d'avoir quelques notions en Python. Nous allons ici en résumer les principaux concepts, sans prétendre vouloir couvrir le langage de manière exhaustive. Nous ne pouvons que vous recommander de lire l'excellente référence Dive into Python, disponible gratuitement sur [2]. Python est un langage de programmation interprété et orienté objet. Il fonctionne selon un typage dynamique fort, dispose d'une gestion automatique de la mémoire et d'un système de gestion d'exceptions. Il est support´e par la plupart des plates-formes informatiques, dont entre autre Mac OS X, Linux et Windows. Il existe de plus une multitude de librairies qui font de Python un langage complet et offrant un développement rapide de scripts en tout genre, mais aussi d'applications complètes.

La syntaxe du langage est conçue pour être extrêmement lisible. C'est pourquoi la ou certains langages utilisent de la ponctuation, python utilise des mots-cl´es anglais. De plus, les blocs sont identifiés par l'indentation et pas par des accolades, tandis que la fin d'une ligne ne comporte aucun caractère spécial

Modules, classes et méthodes

Dans la terminologie Python, un module représente simplement un fichier de code, contenant
une ou plusieurs classes. Il existe deux moyens d'importer des modules externes :
le premier permet simplement d'indiquer `a Python dans quels fichiers chercher les méthodes,
les appels s'effectuant par l'utilisation du préfixe du nom du module. Cette
méthode est mise en place en utilisant : import module

La deuxième méthode permet quant `a elle d'importer directement les classes et méthodes dans l'espace de nom local, permettant ainsi d'éviter l'utilisation du préfixe. De plus, il est possible de spécifier quelles classes nous souhaitons rendre visible. Le code suivant est utilisé pour cela : from MonModule import MaClasse

En Python, tout est objet, que ce soit nos propres classes, des instances de classes, mais aussi les types de base et les fonctions. De plus, Python supporte l'héritage multiple mais ne permet pas de surcharger les opérateurs. Par défaut, une classe met a disposition deux accesseurs pour ses attributs :

getattr(objet, 'nom de l'attribut')
setattr(objet, 'nom de l'attribut', nouvel_attribut)

Une classe se définit grâce au mot-clé class, l'héritage étant ensuite défini en utilisant le ou les noms des classes à hériter entre parenthèses, le tout séparé par des virgules le cas échéant. Afin d'initialiser certaines données membres de la classe, il est possible de définir une méthode d'initialisation, appelée __init__(). Cette méthode n'est pas `a considérer comme un constructeur, car lors de son appel implicite, l'objet est déjà crée. De même, comme il ne s'agit pas d'un constructeur, il convient d'appeler la méthode __init__() du parent le cas échéant. Consultez l'exemple de la figure 1 afin de voir sa syntaxe complète. Les méthodes d'une classe Python ont toutes besoin d'un paramètre afin de se référer à l'objet courant, stock´e dans le premier paramètre d'une fonction. Si nous regardons l'exemple ci-dessous, nous constatons que chaque méthode à comme premier paramètre self, représentant l'instance de la classe. Ce paramètre est automatiquement inséré lors d'appels de méthodes par Python, et son nom n'est qu'une convention, vous pouvez donc le nommer comme bon vous semble.
Notons finalement que pour définir une classe vide, il convient de lui adjoindre le mot-clé pass, qui correspond `a une instruction vide (équivalente à {} en C). Par exemple :

class Afficheur:
def __str__(self):

```
return "Je suis un " + self.__class__.__name__
def __repr__(self):
return repr(self.__str__())
class Forme:
def __init__(self,couleur="blanc"):
self.couleur=couleur
def __str__(self):
return "de couleur " + self.couleur
class Carre(Forme,Afficheur):
def __init__(self, cote=0):
Forme.__init__(self, "rouge")
self.cote = cote
def __str__(self):
return Afficheur.__str__(self) + " " + Forme.__str__(self) +\
" et de cote " + str(self.cote)
class Vide(Forme,Afficheur):
pass
if __name__ == "__main__":
tuple = (Carre(12), Forme(), Carre(), Vide())
for item in tuple:
print item
```

MODE D'EMPLOI

Installation de Django

Deuxièmement, il faut bien évidemment installer Django lui-même. Les différentes versions de Django sont disponibles au lien suivant : https://www.djangoproject.com/download/. Je vous recommande d'utiliser la dernière version officielle qui est au moment de l'écriture de ce rapport la version 1.5.

Si vous avez installé pip plus tôt, vous pouvez installer Django à partir d'une invite de commandes en écrivant : « pip install Django==1.5.1 » (1.5.1 peut être remplacé par une autre version).

Si vous n'avez pas installé pip plus tôt, il vous faudra télécharger l'archive de Django (.tar.gz) manuellement, l'extraire avec un programme comme 7-zip et ouvrir une fenêtre de commande dans le dossier où se trouve le fichier setup.py. Par la suite, il suffit d'un simple : « python setup.py install » pour compiler et installer Django. Django sera installé dans le répertoire suivant : (Chemin vers votre répertoire Python)\Lib\site-packages\django.

Testons maintenant notre installation de Django afin de s'assurer que tout fonctionne. Dans une invite de commandes écrivez python afin d'entré dans un shell Python

puis : « import django » et « django.VERSION »

Configuration d'une base de données

Il ne nous reste plus qu'une seule étape avant de commencer à développer notre site Django, il s'agit de configurer un serveur de base de données pour celui-ci. Django est compatible avec PostgresSQL, SQLite3, MySQL ainsi qu'Oracle.

L'installation de PostgresSQL, MySQL ou Oracle nécessitera en premier lieu d'installer et configurer le serveur de base données puis d'installer la bonne librairie Python selon votre serveur. Pour ce tutoriel nous allons simplement utiliser SQLite3 car il ne nécessite aucune autre configuration.

Commencer un projet

Nous sommes maintenant prêts à créer notre premier site Django. Pour commencer créer un nouveau dossier afin de commencer à travailler et ouvrez une invite de commandes. Nous allons créer notre projet avec la commande suivante : « django-admin.py startproject monsite ». La commande startproject devrait avoir créé l'arborescence suivante :

```
monsite/
        manage.py
        monsite/
                __init__.py
                settings.py
                urls.py
                wsgi.py
```

Voici quelques détails sur les fichiers créé :

- monsite/ : Ce dossier ne sert qu'a contenir votre projet Django, sont nom n'as aucune importance vous pouvez le renommer si vous voulez.
- manage.py : Ce fichier Python contient toutes les commandes nécessaires afin de gérer votre projet.
- monsite/monsite/ : Ce dossier est le paquet Python de votre projet. Son nom est le paquet Python que vous devrez appeler si vous désirez utiliser quelque chose à l'intérieur de celui-ci. (ex: import monsite.settings)
- __init__.py : Ce fichier est requis par Python afin que celui-ci puisse traiter le dossier comme un paquet Python. C'est un fichier vide.
- settings.py : Ce fichier contient les variables de configuration de votre projet Django.
- urls.py : Ce fichier contient les adresses URLs de votre projet.
- wsgi.py : Ce fichier permet à votre serveur web compatible au WSGI à servir votre site.

Conclusion

je tiens tout d'abord à remercier Mr durant , qui m'a permis de travailler sur ce projet aui me tenait tout particulièrement a cœur , par intérêt que je porte à l'informatique et la programmation et deuxièmement au vue de mes connaissance .Le but de ce projet c'est de créer un site web , il m'a permis d'enrichir mes connaissances sur la programmation web et m'a fait découvrir des outils très intéressante .